${\bf Training Data}$

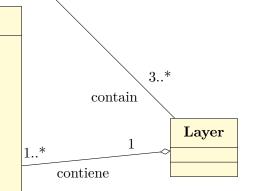
- $m_{training}$ DataFile : ifstream
- + TrainingData(filename : const string)
- $+ \sim TrainingData()$
- + isEof(void) : bool
- + getTopology(&topology : vector<unsigned>) : void
- $+ \hspace{0.1cm} \hspace{0.1cm} + \hspace{0.1cm} \hspace{0.1cm} \hspace{0.1cm} \hspace{0.1cm} \hspace{0.1cm} \hspace{0.1cm} + \hspace{0.1cm} \hspace{0.$
- + getTargetOutputs(&targetOutputVals : vector<double>) : unsigned

Network

- m_layers : vector<Layer>
- m_error : double
- m recentAverageError : double
- $m_recentAverageSmoothingFactor: double$
- + Network(&topology : const vector<unsigned>)
- $+ \sim Network()$
- + feedForward(&inputVals : vector<double>) : void
- + backProp(&targetVals : vector<double>) : void
- + getResults(&resultVals : vector<double>) : void
- + getRecentAverageError(void) : double

Neuron

- eta: double
- alpha: double
- activationFunction: double
- activationFunctionDerivative : double
- randomWeight: void
- sumDOW(&nextLayer : const Layer) : double
- m_outputVal : double
- m_outputWeights : vector<Connection>
- m_myIndex : unsigned
- m_gradient : double
- + Neuron(numOutput : unsigned, myIndex unsigned)
- $+ \sim Neuron()$
- + getOutputVal(void) const : double
- + feedForward(&prevLayer : const Layer) : void
- + calcOutputGradients(targetVal : double) : void
- + calcHiddenGradients(&nextLayer : const Layer) : void
- + updateInputWeights(&nextLayer : Layer) : void



1

holds Connection

1..*

- weight : double
- deltaWeight : double